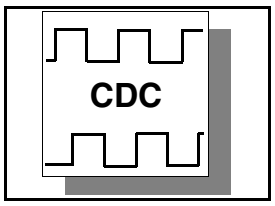
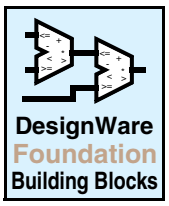




DW_data_sync_1c

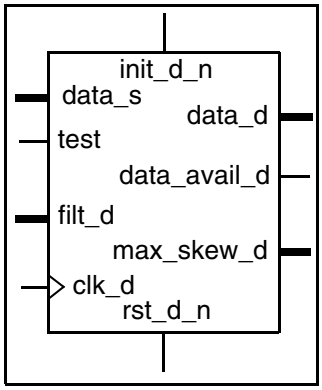
Single Clock Filtered Data Bus Synchronizer

Version, STAR, and myDesignWare Subscriptions: IP Directory

Features and Benefits

Revision History

- Synchronizes on data without incoming clock source
- Parameterized data bus width
- Parameterized number of synchronizing stages
- Parameterized test feature
- All output registered
- Ability to model missampling of data on incoming clock domain



Description

The DW_data_sync_1c can synchronize a data bus to a clock domain when only the data bus is available but not the source clock. The bus is double register synchronized (which is configurable up to thee stages of synchronization) and then monitored for changes. Once a change is detected, the module looks for a specified number of clocks of stability in the data bus before passing the synchronized data bus value out. The number of cycles required to declare the bus stable is specified by the input port `filt_d` and has a range that is determined by the parameter *filt_size*.

This synchronizer is especially well suited for synchronizing a bus from off-chip, where the input skew between bits of the bus may have a large variance due to board-level wiring and the skew also changes from board design to board design. With the possibility of high bit-to-bit skew and a relatively high frequency `clk_d`, this module is the prime candidate for the using a *verif_en* value of 2. Board-level routing skews can create bit-to-bit skews greater than one cycle of a fast `clk_d` signal which leads to the need for modeling missampling across more that a single clock delay.

A unique built-in verification feature allows the designer to turn on a random sampling error mechanism that models skew between bits of the incoming data bus from the source domain (for more information, see "Simulation Methodology" on page 5). This facility provides an opportunity for determining system robustness during the early development phases and without having to develop special test stimulus.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
data_s	width	Input	Source Domain data vector
clk_d	1	Input	Destination Domain clock source
rst_d_n	1	Input	Destination Domain asynchronous reset (active low)
init_d_n	1	Input	Destination Domain synchronous reset (active low)

Table 1-1 Pin Description (Continued)

Pin Name	Width	Direction	Function
filt_d	filt_size	Input	Destination domain filter time specification
test	1	Input	Scan test mode select
data_avail_d	1	Output	Destination domain data update output
data_d	*width*	Output	Destination domain data vector
max_skew_d	*filt_size* + 1	Output	Destination domain maximum skew detected between bits for any `data_s` bus transition

Table 1-2 Parameter Description

Parameter	Values	Description
width	1 to 1024 Default: 8	Vector width of input `data_s` and output `data_d`
f_sync_type	0 to 4 Default: 2	Forward synchronization type Defines type and number of synchronizing stages: ■ 0: single clock design, no synchronizing stages implemented ■ 1: 2-stage synchronization, with first stage negative-edge capturing and second stage positive-edge capturing ■ 2: 2-stage synchronization with both stages positive-edge capturing ■ 3: 3-stage synchronization with all stages positive-edge capturing ■ 4: 4-stage synchronization with all stages positive-edge capturing
filt_size	1 to 8 Default: 1	`filt_d` vector size The width in bits for the `filt_d` vector input
tst_mode	0 or 1 Default: 0	Test Mode ■ 0: no 'latch' is inserted for scan testing ■ 1: insert negative-edge capturing flip-flip on `data_s` input vector when the `test` input is asserted.
verif_en	0 to 4 Default: 0	Verification Enable Control ■ 0: No sampling errors inserted ■ 1: Sampling errors are randomly inserted with 0 or up to 1 destination clock cycle delays ■ 2: Sampling errors randomly inserted with 0, 0.5, 1, or 1.5 destination clock cycle delays ■ 3: Sampling errors are randomly inserted with 0, 1, 2, or 3 destination clock cycle delays ■ 4: Sampling errors randomly inserted with 0 or up to 0.5 destination clock cycle delays For more information about *verif_en*, see “Simulation Methodology” on page 5.

Table 1-3 Synthesis Implementations

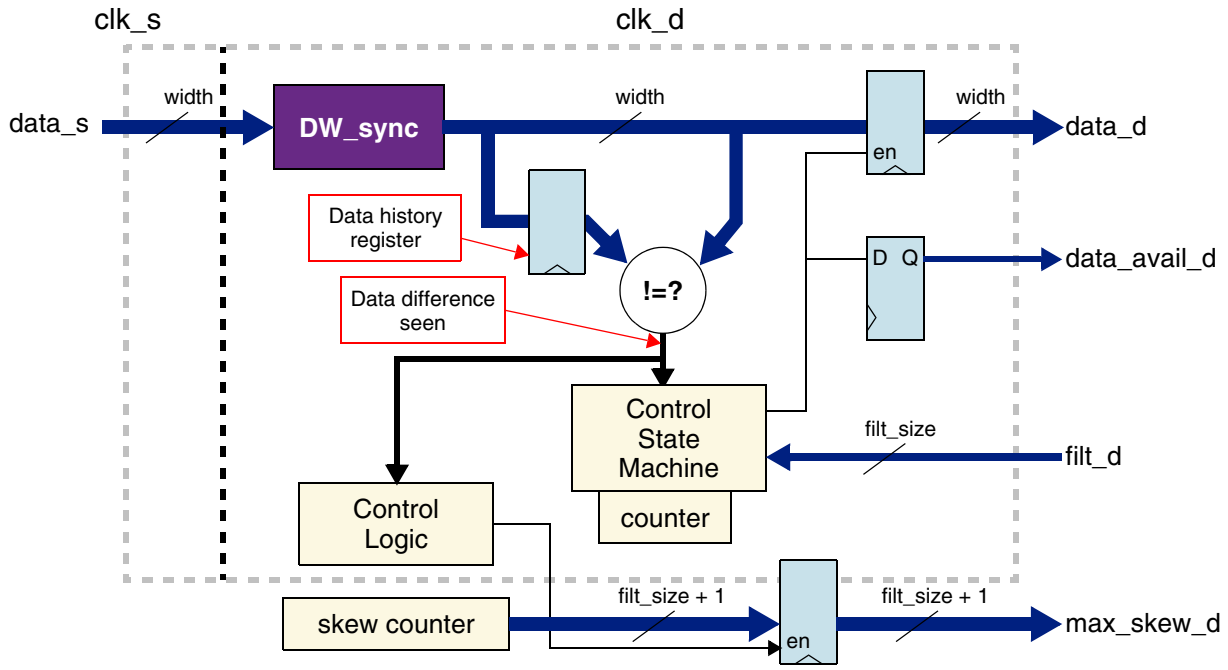
Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

Table 1-4 Simulation Models

Model	Function
DW03.DW_DATA_SYNC_1C_CFG_SIM	Design unit name for VHDL simulation (no missampling)
DW03.DW_DATA_SYNC_1C_CFG_SIM_MS	Design unit name for VHDL simulation (missampling enabled)
dw/dw03/src/DW_data_sync_1c_sim.vhd	VHDL simulation model source code (modeling RTL)
dw/sim_ver/DW_data_sync_1c.v	Verilog simulation model source code

Block Diagram

Figure 1-1 DW_data_sync_1c Basic Block Diagram



filt_d Interpretation

The `filt_d` value specifies the number of `clk_d` cycles in which the `data_d` output is stable after the first occurrence of that `data_d` value. Once stable data is observed based on the `filt_d` value, the `data_d` and `data_avail_d` outputs are registered. For example, if you want to see two consecutive `clk_d` cycles of stable output data from DW_sync, set `filt_d` to 1.

data_s Bus Transition Guidelines as Related to filt_d

To prevent the dropping of data_s values, choosing a filt_d value as it relates to the frequency of data_s transitions and the potential bit skew within data_s in terms of clk_d is important. For the simple case where no skew between bits is expected on data_s in which the bus value is captured in its entirety by clk_d, data_s cannot change more often than every 'filt_d + 1' clk_d cycles and filt_d would typically be set to '1' for optimum throughput.

In the case where there is less than a clk_d cycle skew between bits of data_s but the skew is greater than 0, then data_s cannot change more often than every 'filt_d + 2' clk_d cycles. Again, filt_d can be set as low as '1' for optimum performance without dropping data.

However, in the more severe bit skew cases on data_s of greater than one clk_d cycle, the data_s bus value changes should not be made more often than the number of clk_d cycles based on the formula:

max_skew_d + 2

where max_skew_d is the maximum number of clk_d cycles of data_s bit skew (see immediately below for the detailed description of max_skew_d) and filt_d is recommended to be set to 'max_skew_d + 1'.

max_skew_d Interpretation

max_skew_d is an ongoing result which defines the maximum skew (in clk_d cycles) between bits within data_s per data_s bus transition that has been collected over time. The idea is to use max_skew_d to determine an optimized filt_d value. One approach to determining max_skew_d is run the system through an initialization/test period which is representative of normal behavior for data_s. Then, filt_d can be set to the recommended value of max_skew_d + 1 for max_skew_d values '2' or higher. When max_skew_d is 1, filt_d can be set to '1' for optimal throughput as long as 'data_s bus transition guidelines' are maintained as described above. It is not recommended to set filt_d to less than max_skew_d. If filt_d is set to 0, it implies there is no skew between bits within data_s and, therefore, no skew measurements aree taken. The resulting max_skew_d is 0, in this case.

Simulation Methodology

Because this component contains synchronizing devices, there are two methods available for simulation. One method is to use the simulation models that emulate the RTL model. Or, you can enable modeling of random skew between bits of signals traversing to and from each domain (called missampling).

To use the simulation models that emulate the RTL model, no special configuration is required.

To use missampling requires the following considerations:

- To enable missampling in Verilog simulations, define the macro DW_MODEL_MISSAMPLES:

  ```
  `define DW_MODEL_MISSAMPLES
  ```

  If `DW_MODEL_MISSAMPLES is defined, the *verif_en* parameter comes into play to configure the simulation model as described by Table 1-2 on page 2. If `DW_MODEL_MISSAMPLES is not defined, the Verilog simulation model behaves as if *verif_en* is set to 0.

- To enable missampling in VHDL simulations, a simulation architecture named sim_ms is provided. The parameter *verif_en* only has meaning when using sim_ms. That is, when the `sim` simulation architecture is used instead, the model behaves as though *verif_en* is set to 0. For examples of how each architecture is used, see "HDL Usage Through Component Instantiation - VHDL" on page 8.

Suppressing Warning Messages During Verilog Simulation

The Verilog simulation model includes macros that allow you to suppress warning messages during simulation.

To suppress all warning messages for all DWBB components, define the DW_SUPPRESS_WARN macro in either of the following ways:

- Specify the Verilog preprocessing macro in Verilog code:

  ```
  `define DW_SUPPRESS_WARN
  ```

- Or, include a command line option to the simulator, such as:

  `+define+DW_SUPPRESS_WARN` (which is used for the Synopsys VCS simulator)

The warning messages for this model include the following:

- If values other than 1 or 0 are present on a clock port, the following message is displayed:

  ```
  WARNING: <instance_path>.<clock_name>_monitor:
      at time = <timestamp>, Detected unknown value, x, on <clock_name> input.
  ```

  To suppress only this warning message for all DWBB components, use the following macro:

  - Define the DW_DISABLE_CLK_MONITOR macro. You can define this macro in the following ways:
    - Specify the Verilog preprocessing macro in Verilog code:

      ```
      `define DW_DISABLE_CLK_MONITOR
      ```

    - Or, include a command line option to the simulator, such as:

      `+define+DW_DISABLE_CLK_MONITOR` (which is used for the Synopsys VCS simulator)

  This message is also suppressed using the DW_SUPPRESS_WARN macro explained earlier.

Timing Diagrams

Figure 1-2 depicts the case in which data_s contains on intra-bit skew meaning all bits arrive into the component at exactly the same time. The important concept to note here is that since filt_d is 1, the number of clk_d cycles of stable data on the internal signal dw_sync_data_d[8:0] must 2 before data is considered valid. 2 clk_d cycles defined 1 cycle when first observed at the DW_sync output and 1 cycle (as defined by filt_d = 1), which has the same value immediately after the first observed occurrence. Note that in the diagram, signal dw_sync_data_d[8:0] is provided as reference only and it represents the data_d of the instance of DW_sync contained in DW_data_sync_1c.

Figure 1-2 Fundamental Case where No Intra-bit Skew on data_s

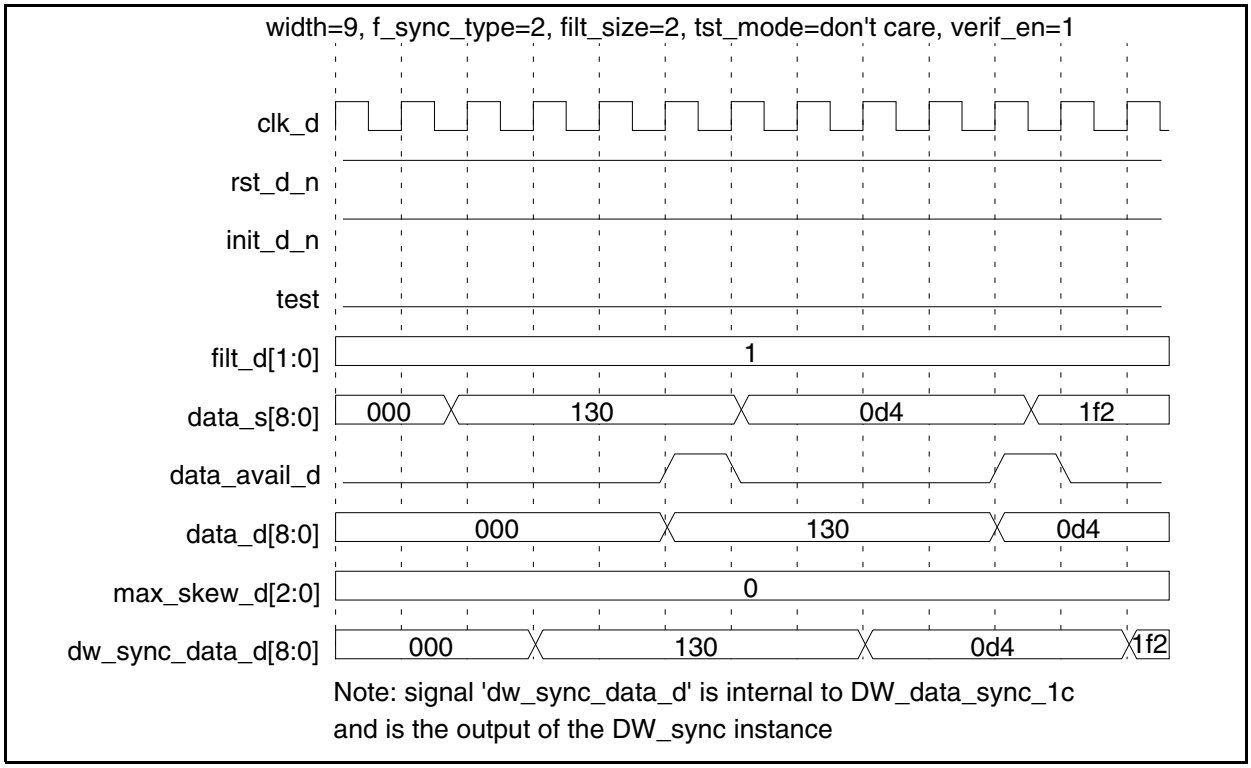
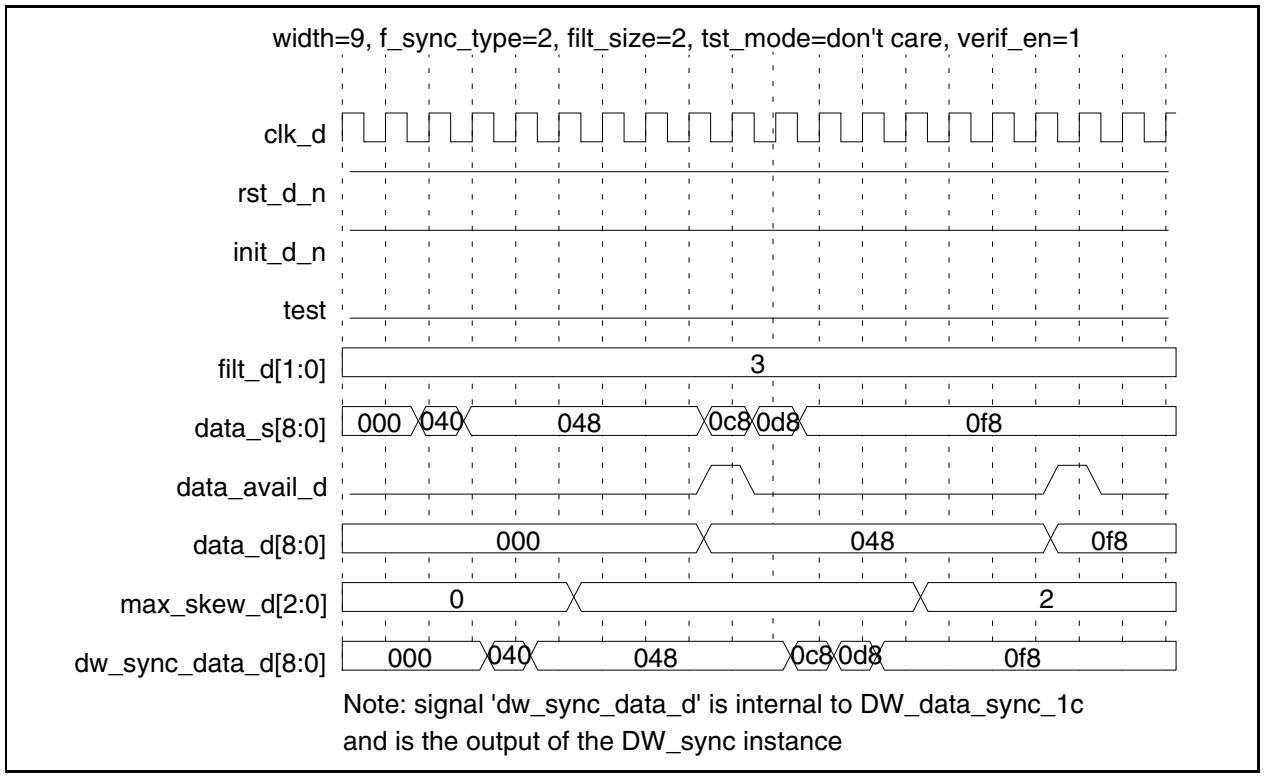


Figure 1-3 shows an example of how intra-bit skew on data_s as seen by the destination domain (clk_d) contributes to the result of the output max_skew_d. For illustration purposes, the intra-bit skew on data_s is grossly exaggerated to clearly cross clk_d sampling boundaries. At beginning of time, bits on data_s arrive as all 0s then change to 0x040 then to 0x048 effectively one clk_d sampling later. The value of 0x048 settles long enough (satisfying filt_d) for data_avail_d to activate and the captured data_s value of 0x048 to be presented to the destination domain at that time. Meanwhile, due to the data_s intermediate changes before settling (once after the previously settled value of all 0s) a new max_skew_d value is calculated to be 1.

In the subsequent data_s value changes after 0x048 and before settling on 0x0f8 the max_skew_d value gets updated to 2 because data_s underwent intra-bit skew across two clk_d cycles as it arrived into the component as 0x0c8, then 0x0d8, then finally settling at 0x0f8.

Figure 1-3 Intra-bit Skew on data_s Affecting max_skew_d Results



Related Topics

- Memory – Registers Overview
- DesignWare Building Block IP User Guide

HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DW_Foundation_comp.all;

entity DW_data_sync_1c_inst is
      generic (
            inst_width : INTEGER := 8;
            inst_f_sync_type : INTEGER := 2;
            inst_filt_size : INTEGER := 1;
            inst_tst_mode : INTEGER := 0;
            inst_verif_en : INTEGER := 2
            );
      port (
            inst_clk_d : in std_logic;
            inst_rst_d_n : in std_logic;
            inst_init_d_n : in std_logic;
            inst_data_s : in std_logic_vector(inst_width-1 downto 0);
            inst_filt_d : in std_logic_vector(inst_filt_size-1 downto 0);
            inst_test : in std_logic;
            data_avail_d_inst : out std_logic;
            data_d_inst : out std_logic_vector(inst_width-1 downto 0);
            max_skew_d_inst : out std_logic_vector(inst_filt_size downto 0)
            );
    end DW_data_sync_1c_inst;


architecture inst of DW_data_sync_1c_inst is
begin

    -- Instance of DW_data_sync_1c
    U1 : DW_data_sync_1c
    generic map ( width => inst_width, f_sync_type => inst_f_sync_type,
                filt_size => inst_filt_size, tst_mode => inst_tst_mode,
                verif_en => inst_verif_en )
    port map ( clk_d => inst_clk_d, rst_d_n => inst_rst_d_n,
                init_d_n => inst_init_d_n,data_s => inst_data_s,
                filt_d => inst_filt_d,test => inst_test,
                data_avail_d => data_avail_d_inst, data_d => data_d_inst,
                max_skew_d => max_skew_d_inst );
end inst;

-- Configuration for use with a VHDL simulator
-- pragma translate_off
library DW03;
configuration DW_data_sync_1c_inst_cfg_inst of DW_data_sync_1c_inst is
  for inst
    -- NOTE: If desiring to model missampling, uncomment the following
```

```
    -- line.  Doing so, however, will cause inconsequential errors
    -- when analyzing or reading this configuration before synthesis.
    -- for U1 : DW_data_sync_1c use configuration DW03.DW_data_sync_1c_cfg_sim_ms;  end
for;
  end for; -- inst
end DW_data_sync_1c_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_data_sync_1c_inst( inst_clk_d, inst_rst_d_n, inst_init_d_n, inst_data_s,
inst_filt_d,
                                                        inst_test,  data_avail_d_inst,
data_d_inst, max_skew_d_inst );

parameter width = 8;
parameter f_sync_type = 2;
parameter filt_size = 1;
parameter tst_mode = 0;
parameter verif_en = 1;


input inst_clk_d;
input inst_rst_d_n;
input inst_init_d_n;
input [width-1 : 0]     inst_data_s;
input [filt_size-1 : 0] inst_filt_d;
input inst_test;
output                       data_avail_d_inst;
output [width-1 : 0] data_d_inst;
output [filt_size : 0] max_skew_d_inst;

    // Instance of DW_data_sync_1c
    DW_data_sync_1c #(width, f_sync_type, filt_size, tst_mode, verif_en)
          U1 ( .clk_d(inst_clk_d), .rst_d_n(inst_rst_d_n),  .init_d_n(inst_init_d_n),
                  .data_s(inst_data_s), .filt_d(inst_filt_d), .test(inst_test),
                  .data_avail_d(data_avail_d_inst), .data_d(data_d_inst),
                  .max_skew_d(max_skew_d_inst) );

endmodule
```

Revision History

For notes about this release, see the *DesignWare Building Block IP Release Notes*.

For lists of both known and fixed issues for this component, refer to the STAR report.

For a version of this datasheet with visible change bars, click here.

Date	Release	Updates
July 2020	DWBB_201912.5	■ Adjusted the material in "Simulation Methodology" on page 5 ■ Adjusted content and title of "Suppressing Warning Messages During Verilog Simulation" on page 5, added the DW_SUPPRESS_WARN macro, and moved it to be near the "Simulation Methodology" on page 5 section
October 2019	DWBB_201903.5	■ Added the "Disabling Clock Monitor Messages" section ■ Added this Revision History table and the document links on this page

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.

All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
www.synopsys.com